**Arrays and Loops**

* Once an array is created, you can fill the elements in an array using loops.
* For example:

int[] a = new int[100];  
for (int i = 0; i < 100; i++)

{

a[i] = i; // fills the array with numbers 0 to 99

}

**Array Length**

* To find the number of elements of an array, use ***array*.length**.
* For example:

for (int i = 0; i < **a.length;** i++)

System.out.println(a[i]);

* Once you create an array, you cannot change its size.
* However, you can
  + change an individual array element
  + create a new array with a bigger size, and copy over the contents from the old array to the new array
* If you frequently need to expand the size of an array while your program is running, you should use a different data structure called an ***array list*.** (See Chapter 5 for more on array lists.)

**Array Out of Bounds Exception**

* Java has bounds checking.
* If you construct an array with 100 elements and then try to access the element a[100] (or any other index outside the range from 0 to 99), your program will terminate with an “array index out of bounds” exception.

**The “for each” Loop**

* Java has a powerful looping construct that allows you to loop through each element in an array (or any other collection of elements) without having to fuss with index values.

The ***enhanced***for loop

for (*variable* : *collection*) *statement*

sets the given *variable* to each element of the *collection* and then executes the *statement* (which, of course, may be a block of statements).

Note that the loop *variable* of the “for each” loop traverses the elements of the array, not the index values.

* The *collection* expression must be an array or an object of a class that implements the Iterable interface, such as ArrayList.
* We discuss array lists in Chapter 5 and the Iterable interface in Chapter 9.

For example:

for (int element : a)

System.out.println(element);

prints each element of the array a on a separate line.

You should read this loop as “**for each element in a**”.

The designers of the Java language considered using keywords, such as foreach and in.

But this loop was a late addition to the Java language, and in the end, nobody wanted to break the old code that already contained methods or variables with these names (such as System.in).